WaterBot

Infrastructure Cost Analysis & Migration Guide

By Ram Harikrishnan

February 2026

Arizona State University — Computer Engineering

Executive Summary

The current AWS setup costs an estimated $300–370/month to run a chatbot that realistically needs about $15–40/month of compute. The overspend is driven by enterprise-grade defaults — Multi-AZ RDS, 2–4 Fargate tasks, NAT Gateways, and an Application Load Balancer — applied to a project that serves a small number of concurrent users. Moving to Railway, Render, or a self-hosted VPS would cut the bill by 85–95% with zero loss of capability at this scale.

1. Current AWS Cost Breakdown

1.1 Resource Inventory

The following resources are defined in the CDK stack (iac/cdk/stacks/app_stack.py) and are currently running in AWS us-east-1:

Resource	Specification	Purpose
ECS Fargate	2 tasks min / 4 max — 2 vCPU + 4 GB RAM each	Application container hosting
RDS PostgreSQL	t3.small, Multi-AZ, 20 GB storage, 30-day backups	Messages + vector embeddings (pgvector)
Application LB	Standard ALB with custom-header authentication	Routes traffic to Fargate tasks
VPC (2 private AZs)	max_azs=2, private subnets with NAT egress	Network isolation for Fargate + RDS
NAT Gateway x2	One gateway per availability zone	Outbound internet for private containers

Resource	Specification	Purpose
CloudFront	CDN distribution with CloudFront Function basic auth	Static caching + HTTPS termination
DynamoDB	Pay-per-request billing mode	User ratings and feedback
S3 (3 buckets)	Export, transcripts, Postgres backups	Durable object storage
Lambda (3 functions)	dynamo export, db_init, postgres_backup	Scheduled automation tasks
Secrets Manager (3)	OpenAI key, DB credentials, Basic Auth token	Secure secret injection at runtime
ECR	Single-image container registry	Docker image storage for CI/CD
CloudWatch Logs	Container + RDS log streams (non-blocking)	Operational observability
EventBridge + SQS x2	Scheduled rules + dead-letter queues	Backup scheduling with failure handling

1.2 Monthly Cost Estimate (us-east-1, February 2026)

Service	Monthly Cost	Notes
ECS Fargate (2 tasks x 2 vCPU x 4 GB, 24/7)	**~$142**	$0.04048/vCPU-hr + $0.004445/GB-hr
RDS t3.small Multi-AZ	**~$100**	Multi-AZ doubles the ~$50 single-AZ cost
NAT Gateways x2	**~$65**	$0.045/hr each; required for private subnets
Application Load Balancer	**~$18**	$0.0225/hr fixed + LCU usage charges
CloudWatch Logs	**~$8**	Log ingestion and retention
Secrets Manager	**~$2**	$0.40/secret/month x 3 secrets
S3 (3 buckets)	**~$3**	Storage + requests at low traffic levels
DynamoDB	**~$2**	Pay-per-request at low traffic levels
CloudFront	**~$2**	Low traffic — mostly within free tier
ECR	**~$1**	Single image, minimal storage footprint
Lambda + EventBridge + SQS	**~$0**	All well within AWS free tier limits
TOTAL	**~$343 / month**	**~$4,116/year — excludes data transfer (+$10–30/mo)**

Current AWS spend: ~$343/month ($4,116/year). Actual compute need for this workload: ~$15–40/month. Overspend ratio: approximately 10x.

2. Why This Is Overkill

2.1 Two Fargate Tasks Running 24/7 | Overspend: ~$71/month

The CDK stack sets desired_count=2 with auto-scaling up to 4 tasks. Each task carries 2 vCPU and 4 GB RAM. WaterBot is a FastAPI async application — a single task with 1 vCPU and 1 GB RAM can handle hundreds of concurrent users without performance issues. The second task exists for high-availability reasons that only matter at scale.

- Reducing desired_count to 1 immediately halves the Fargate line item.
- A single async FastAPI worker can serve 100–200 concurrent connections on 1 vCPU.
- The second idle task wastes ~$71/month of compute 24 hours a day.

2.2 Multi-AZ RDS for a Non-Critical Workload | Overspend: ~$50/month

Multi-AZ RDS runs an identical standby replica in a second availability zone and automatically fails over if the primary goes down. Failover takes 60–120 seconds. For a chatbot that stores conversation messages — not financial transactions — this level of availability is unnecessary. A single-AZ t3.micro or t3.small with daily backups is entirely sufficient. The standby replica adds ~$50/month with no user-visible benefit at this scale.

2.3 NAT Gateways | $65/month to Give Containers Internet Access

The VPC places Fargate and RDS in private subnets — a secure default. But private subnets require NAT Gateways for outbound internet access (OpenAI API, Bedrock, etc.), and NAT Gateways cost $0.045/hour each. With two availability zones that is $64.80/month — more than many competing platforms charge for an entire production-grade hosted application.

- Alternative: use public subnets with tight security group rules. Marginally less 'enterprise' but perfectly secure for a public-facing chatbot — and the cost drops to $0.

2.4 Application Load Balancer for a Single Service | Overspend: ~$18/month

The ALB routes traffic to the ECS service and enables the X-Custom-Header security check. At this scale, a simple reverse proxy (nginx or Caddy) in front of the container performs the same function at no infrastructure cost. The custom header check can be replicated with a single middleware line.

2.5 DynamoDB for Ratings Storage

DynamoDB is pay-per-request at this scale so the direct cost is minimal (~$2/month). However it adds operational complexity — separate IAM policies, a separate SDK client, and separate backup logic — for a ratings table that could trivially be a second table in the existing

PostgreSQL database. Consolidating into Postgres reduces code surface area and removes an entire AWS service dependency.

Waste Source	Monthly Overspend	Fix
2nd Fargate task (idle)	~$71	Set desired_count=1
Multi-AZ RDS standby replica	~$50	Switch to single-AZ; add daily snapshots
NAT Gateways x2	~$65	Move to public subnets + security groups
Application Load Balancer	~$18	Replace with nginx/Caddy reverse proxy
DynamoDB ratings table	~$2	Consolidate into existing PostgreSQL
Total Removable Overspend	**~$206/month**	**Right-sized AWS: ~$60–80/month**

3. Alternative Platform Comparison

All four options below support the existing WaterBot Dockerfile with no code changes. AWS credentials for Bedrock Knowledge Base are passed as environment variables on any platform

Option A — Railway (~$10–20/month)

Railway runs Docker containers directly and includes managed PostgreSQL. WaterBot's existing Dockerfile deploys with zero code changes — update DATABASE_URL and deploy. GitHub integration, automatic SSL, and connection pooling are all included.

Component	Solution	Cost
FastAPI App	Railway service (1 vCPU, 512 MB–1 GB RAM, auto-deploy from Dockerfile)	~$5–10/mo
PostgreSQL	Railway managed Postgres with built-in connection pooling	~$5–10/mo
Static Files/CDN	Cloudflare free tier as a proxy	$0
Secrets	Railway environment variables (encrypted at rest)	$0
TOTAL		**~$10–20/mo**

- Pros: Zero code changes; GitHub push-to-deploy; managed Postgres; automatic SSL; Bedrock works via env vars.
- Cons: No built-in multi-region; occasional cold starts on Hobby plan (solvable with Pro tier).

Option B — Render (~$14/month)

A render.yaml was previously committed to the WaterBot repository, indicating this platform was evaluated before. Render is a traditional PaaS with persistent disks, always-on services (no sleep on paid plans), and a built-in global CDN.

Component	Solution	Cost
FastAPI App	Render Web Service — Starter: 512 MB RAM, always-on	**$7/mo**
PostgreSQL	Render managed Postgres — Starter: 256 MB RAM, 1 GB storage	**$7/mo**
CDN	Render includes a global CDN on all web services by default	**$0**
TOTAL (Starter)	**Standard tier: $25 DB + $25 service = $50/mo**	**~$14/mo**

- Pros: Persistent disk; always-on; GitHub auto-deploy; built-in health checks. Project already has familiarity with this platform.
- Cons: Starter Postgres is resource-constrained (256 MB RAM). Standard tier brings it to $45–50/month.

Option C — Supabase + Fly.io (~$36/month)

Supabase provides a generous hosted PostgreSQL service; Fly.io hosts the container with excellent multi-region Anycast networking. This combination is the best choice if the project anticipates growth or wants Supabase's Auth/Realtime features later.

Component	Solution	Cost
FastAPI App	Fly.io — shared-cpu-2x, 512 MB RAM, persistent volumes available	**~$11/mo**
PostgreSQL	Supabase Pro — 8 GB DB, 100 GB storage, 250 GB bandwidth, dashboard	**$25/mo**
CDN	Cloudflare free tier	**$0**
TOTAL	**Two platforms to manage; Supabase Auth/Realtime available if needed later**	**~$36/mo**

Option D — Self-Hosted VPS (Hetzner) (~$8–10/month)

Run the full stack with Docker Compose on a single Hetzner CX22 VPS (2 vCPU, 4 GB RAM, 40 GB SSD, ~$6.50/month). FastAPI, PostgreSQL, and Caddy all run on the same machine — zero network latency between app and database. The existing docker-compose.yml in the WaterBot repository already provides the foundation.

Component	Solution	Cost
VPS	Hetzner CX22 — 2 vCPU, 4 GB RAM, 40 GB SSD (EU-based, GDPR-friendly)	**~$6.50/mo**
Database	PostgreSQL in Docker Compose on same machine — zero network overhead	**Included**
Web Server	Caddy auto-HTTPS + reverse proxy — handles SSL certificates automatically	**$0**
Backups	Hetzner automated daily snapshots	**~$1.50/mo**
CDN	Cloudflare free tier as front-end proxy	**$0**
TOTAL	**Single async FastAPI process handles 100–200 concurrent users on 1 vCPU**	**~$8–10/mo**

- Pros: Cheapest option. Full control. No vendor lock-in. No network latency between app and database. Hetzner has excellent uptime.
- Cons: You own the operations: OS patching, Postgres upgrades, disk management, and monitoring. No managed failover if the VPS goes down (typically 2–5 min Hetzner SLA).

4. Side-by-Side Platform Comparison

Criteria	Current AWS	Railway	Render	Supabase+Fly.io	Hetzner VPS
Monthly Cost	~$343	~$15	~$14–45	~$36	~$10
Annual Cost	~$4,116	~$180	~$168–540	~$432	~$120
Migration Effort	—	Low	Low	Medium	Medium
Ops Burden	Low (managed)	Low	Low	Low	High
Scalability	Excellent	Good	Good	Good	Manual
HA / Failover	Enterprise	Basic	Basic	Basic	None
Postgres	RDS managed	Managed	Managed	Supabase	Self-managed
Docker Deploy	ECR + ECS	Dockerfile	Dockerfile	Fly.io	Compose
Domain + SSL	CloudFront	Built-in	Built-in	Built-in	Caddy
Bedrock KB	Native	Env vars	Env vars	Env vars	Env vars

* Bedrock Knowledge Base is an AWS-only API — but it is just a boto3 call. On any non-AWS host, pass AWS_ACCESS_KEY_ID, AWS_SECRET_ACCESS_KEY, and AWS_KB_ID as environment variables. No infrastructure change required.

5. What You Would Lose Moving Off AWS

Feature	Impact	Mitigation
Bedrock Knowledge Base	None	It's an API call, not infra. Pass AWS credentials as environment variables.
Multi-AZ RDS failover	60–120s if DB host fails	Nightly snapshots + restore is sufficient for this workload.
ECS auto-scaling	Cannot add containers under load	A single well-sized container handles this traffic. No scaling needed.
CloudWatch Logs	Lose AWS-centralised logging	Papertrail, Logtail, or self-hosted Loki have free tiers.
ALB custom header auth	Lose X-Custom-Header layer	Replace with CloudFront basic auth or a simple reverse-proxy middleware rule.

6. Recommendations

6.1 Short-Term: Move to Railway (~$15/month)

Deploy the existing Dockerfile to Railway. Change DATABASE_URL to Railway's managed Postgres connection string. Keep the same AWS_ACCESS_KEY_ID, AWS_SECRET_ACCESS_KEY, and AWS_KB_ID environment variables for Bedrock calls. Zero code changes required — this is a pure deployment and environment variable change.

Estimated annual saving: **$3,936** ($343 - $15) x 12 months

6.2 Long-Term: Hetzner VPS + Docker Compose (~$10/month)

Set up a Hetzner CX22 with Docker Compose, Caddy for automatic HTTPS, and a cron-based backup to S3 or Backblaze B2. The existing docker-compose.yml already provides the application stack — the remaining work is OS-level setup (one-time, ~2 hours). PostgreSQL running on the same machine eliminates all network latency between the app and database.

- Migration effort: ~2–3 hours for initial setup; ongoing OS patching is the only operational overhead.
- The WaterBot docker-compose.yml already exists in the repo — minimal additional configuration needed.
- Estimated annual saving: $3,996 vs current AWS spend.

6.3 If the Project Scales Significantly

If WaterBot grows to thousands of daily active users and requires an uptime SLA, revisit AWS with a right-sized architecture: one Fargate task (not two), single-AZ RDS (not Multi-AZ), public subnets (no NAT Gateways), and CloudFront pointing directly to the container (no ALB). That configuration reduces the AWS bill to approximately $60–80/month while retaining the full benefits of the AWS ecosystem.

7. Annual Savings Summary

Platform	Monthly Cost	Annual Saving vs. Current AWS
Current AWS (baseline)	**~$343/mo**	—
Railway	~$15/mo	**$3,936 / year**
Render (Starter tier)	~$14/mo	**$3,948 / year**
Supabase + Fly.io	~$36/mo	**$3,684 / year**
Self-Hosted Hetzner VPS	~$10/mo	**$3,996 / year**
Right-Sized AWS (1 task, single-AZ)	~$70/mo	**$3,276 / year**

About This Report

This analysis is based on the WaterBot repository at github.com/S-Carradini/waterbot, the CDK stack definition in iac/cdk/stacks/app_stack.py, and AWS public pricing for us-east-1 as of February 2026. Cost figures for Railway, Render, Supabase, Fly.io, and Hetzner are based on their published pricing pages as of the same date. All figures are estimates; actual costs may vary based on usage patterns, egress traffic, and future pricing changes.